Documentation for opengl2.h and opengl2.c

Steven Andrews, © 2003
See the document "LibDoc" for general information about this and other libraries.

```
void gl2DrawBox(float xlo,float xhi,float ylo,float yhi,float zlo,float zhi);
int gl2State(int state);
void gl2Initialize(float xlo,float xhi,float ylo,float yhi,float
     zlo,float zhi);
```

Requires: `<math.h>`, `<stdio.h>`, `"gl.h"`, `"glut.h"`, `"opengl2.h"`
Example program: `smoldyn.c`

History: Started 12/02, modified substantially and documented 7/03. Some
testing. Modified `KeyPush` 9/17/03.

These routines provide easy use of the OpenGL graphics libraries for the situation in which an object is to be viewed and reoriented by the user. This library was written specifically for a few uses by the *Smoldyn* program and is thus likely to be changed significantly in the near future as I encounter more graphical needs.

Because of the way OpenGL is designed, the standard way of passing of information from one function to another with function arguments is often impossible. This requires the use of many global variables, despite the fact that their use is typically not a good programming practice. The global variables used within this library are listed below. It is suggested that the entire program state of the main program is encapsulated in a single data structure, which is made global, allowing its access by graphing and animation routines.

The way to use this library is to first call `gl2Initialize` with the outside edges of the object to be shown. Then, call `glutDisplayFunc` with the call-back function that is supposed to render the scene (often called `RenderScene`). Then, call `glutTimerFunc` with the call-back function that executes the events that are animated (often called `TimerFunction`). Then, turn control over to OpenGL by calling `glutMainLoop`. From here on, OpenGL calls back to the scene rendering function for redraws and to the timer function for animation and other program execution. OpenGL also calls back to some internal routines in this library for window size changes and key presses.

When the object displayed is 1 or 2 dimensional, the only key events that are monitored are the space bar and the letter 'Q'. The space bar toggles the mode of the gl2state between 0 and 1 and 'Q' puts it irreversibly into state 2. The intent is that the main program is in pause mode when that state is 1, continues when it equals 0, and quits in a normal fashion when it equals 2, although the execution of this functionality is up to the main program. Three dimensional objects allow space bar key events, 'Q' events, and also rotation controls. For the latter, the arrow keys spin the object, the '=' and '-' keys zoom in and out, the 'x', 'X', 'y', 'Y', 'z', and 'Z' keys rotate the object in counterclockwise or clockwise directions about the *x*, *y*, and *z* axes respectively, and the '0' key resets total rotation and zooming to default values.

Externally accessible functions

`gl2DrawBox` draws a wire frame box using the dimensions given. For two dimensions, set `zlo` and `zhi` equal to each other; for one dimension do the same and also set `ylo` and `yhi` equal to each other (good default values are 0, although that's not required). This function does not worry about winding directions.

`gl2State` allows a program to set and access a state variable that indicates whether the program should be in pause mode or quit mode. Also, the user can control the state by pressing the space bar – each press toggles the state to or from the pause mode; 'Q' sets the state irreversibly in the quit mode. With this routine, an argument that is equal to 0 or greater sets that state to that value, while a negative argument has no effect. Either way, the current state is returned. A state of 0 indicates normal operation, a state of 1 indicates pause mode, and a state of 2 indicates quit mode.

`gl2Initialize` allows a program to set up a generic OpenGL output window with minimal effort. Arguments are the outside dimensions of the object; the clipping volume is created so the object can be rotated to any angle without poking outside the volume (using the arrow keys; rotation with other keys can lead to the object escaping). For 1 or two dimensional rendering, enter values of 0 for the excess dimensions.

Internal variables and functions

```
GLfloat ClipSize,ClipMidx,ClipMidy,ClipMidz;
GLfloat ClipLeft,ClipRight,ClipBot,ClipTop,ClipBack,ClipFront;
GLfloat FieldOfView,Near,Aspect;
int Gl2PauseState,Dimension;

void SetupRC(void);
void ChangeSize(int w,int h);
void KeyPush(unsigned char key,int x,int y);
void SpecialKeyPush(int key,int x,int y);
```

Variables

`ClipSize`	length of edge of clipping cube, sized so object can spin.
`ClipMidx`	middle of object on x axis.
`ClipMidy`	middle of object on y axis.
`ClipMidz`	middle of object on z axis.
`ClipLeft`	left edge of clipping cube.
`ClipRight`	right edge of clipping cube.
`ClipBot`	bottom edge of clipping cube.
`ClipTop`	top edge of clipping cube.
`ClipBack`	back edge of clipping cube (z more negative; away from viewer).
`ClipFront`	front edge of clipping cube (z less negative; close to viewer).
`FieldOfView`	field of view on y direction for 3 dimensional, in degrees.
`Near`	distance from viewer to front side of clipping cube.
`Aspect`	aspect ratio of window.
`Gl2PauseState`	pause state, with 0 for continue, 1 for pause.
`Dimension`	dimensionality of object, from 1 to 3.

<u>Functions</u>

`SetupRC` is not a call-back function, but it sets up the OpenGL rendering state. It just sets the background color for the drawing window. This code might be better off in `gl2Initialize`, rather than in a subroutine.

`ChangeSize` is a call-back function. It is called when the user changes the window size or shape, as well as upon initiallization. The function takes in the current window width and height in pixels, in `w` and `h`, respectively. Using those, it sets the clipping volume in such a way that changing the window shape does not distort the image.

`KeyPush` is a call-back function which is called when a key is pressed (but not arrows, etc.) It is set up to rotate the object about the object axes, based on user key pushes, as well as to zoom in and out and reset to a default view.

`SpecialKeyPush` is a call-back function which is called when a special key is pressed (arrows, etc.) It is set up to rotate the object about the viewing axes, based on user key pushes.